CSE 421

Max Flow Min Cut Problem

Yin Tat Lee
Network Flow Applications

Max flow and min cut.
- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

Nontrivial applications / reductions.
- Data mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
Minimum s-t Cut Problem

Given a directed graph $G = (V, E)$ = directed graph and two distinguished nodes: $s =$ source, $t =$ sink.

Suppose each directed edge $e$ has a nonnegative capacity $c(e)$

**Goal:** Find a cut separating $s, t$ that cuts the minimum capacity of edges.
s-t cuts

Def. An s-t cut is a partition \((A, B)\) of \(V\) with \(s \in A\) and \(t \in B\).

Def. The capacity of a cut \((A, B)\): \(\text{cap}(A, B) = \sum_{(u,v): u \in A, v \in B} c(u, v)\)
s-t cuts

Def. An s-t cut is a partition \((A, B)\) of \(V\) with \(s \in A\) and \(t \in B\).

Def. The capacity of a cut \((A, B)\): 
\[
cap(A, B) = \sum_{(u,v): u \in A, v \in B} c(u, v)
\]

\[
\text{Capacity} = 9 + 15 + 8 + 30 = 62
\]
Minimum s-t Cut Problem

Given a directed graph $G = (V, E) = \text{directed graph}$ and two distinguished nodes: $s = \text{source}$, $t = \text{sink}$. Suppose each directed edge $e$ has a nonnegative capacity $c(e)$. 

Goal: Find a s-t cut of minimum capacity

![Graph with s-t cut and capacities](image)

Capacity $= 10 + 8 + 10 = 28$
s-t Flows

Def. An s-t flow is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Def. The value of a flow $f$ is: $\nu(f) = \sum_{e \text{ out of } s} f(e)$

![Graph showing flow values and capacities]

Value = 4
s-t Flows

Def. An **s-t flow** is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$  \hspace{2cm} (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$  \hspace{2cm} (conservation)

Def. The value of a flow $f$ is: $\nu(f) = \sum_{e \text{ out of } s} f(e)$

![Graph with labels and values]

Value = 24
Maximum s-t Flow Problem

Goal: Find a s-t flow of largest value.

Value = 28
Flow value lemma. Let $f$ be any flow, and let $(A, B)$ be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving $s$.

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$
Proof of Flow value Lemma

Flow value lemma. Let $f$ be any flow, and let $(A, B)$ be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving $s$.

$$
\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = \nu(f)
$$

Proof.

$$
\nu(f) = \sum_{e \text{ out of } s} f(e)
$$

By conservation of flow, all terms except $v=s$ are 0

$$
= \sum_{v \in A} \left( \sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)
$$

All contributions due to internal edges cancel out

$$
= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)
$$
Weak Duality of Flows and Cuts

**Flow value lemma.** Let $f$ be any flow, and let $(A, B)$ be any s-t cut. Then the value of the flow is at most the capacity of the cut.

$$v(f) \leq \text{cap}(A, B)$$
Weak Duality of Flows and Cuts

Flow value lemma. Let $f$ be any flow, and let $(A, B)$ be any s-t cut. Then the value of the flow is at most the capacity of the cut.

$$v(f) \leq \text{cap}(A, B)$$

Proof.

\[
v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
\leq \sum_{e \text{ out of } A} f(e) \\
\leq \sum_{e \text{ out of } A} c(e) = \text{cap}(A, B)
\]
Certificate of Optimality

**Corollary:** Suppose there is a s-t cut (A,B) such that
\[ v(f') = \text{cap}(A, B) \]
Then, f is a maximum flow and (A,B) is a minimum cut.

\[ v(f) = 28, \ \text{cap}(A,B) = 28 \]
A Greedy Algorithm for Max Flow

• Start with $f(e) = 0$ for all edge $e \in E$.
• Find an $s$-$t$ path $P$ where each edge has $f(e) < c(e)$.
• Augment flow along path $P$.
• Repeat until you get stuck.
A Greedy Algorithm for Max Flow

• Start with \( f(e) = 0 \) for all edge \( e \in E \).
• Find an \( s \)-\( t \) path \( P \) where each edge has \( f(e) < c(e) \).
• Augment flow along path \( P \).
• Repeat until you get stuck.
Residual Graph

Original edge: \( e = (u, v) \in E \).
- Flow \( f(e) \), capacity \( c(e) \).

Residual edge.
- "Undo" flow sent.
- \( e = (u, v) \) and \( e^R = (v, u) \).

Residual capacity:

\[
c_f(e) = \begin{cases} 
c(e) - f(e) & \text{if } e \in E \\
f(e) & \text{if } e^R \in E
\end{cases}
\]

Residual graph: \( G_f = (V, E_f) \).
- Residual edges with positive residual capacity.
- \( E_f = \{ e : f(e) < c(e) \} \cup \{ e^R : f(e) > 0 \} \).
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Find Path
Ford-Fulkerson Alg: Greedy on $G_f$

$G$:

$G_f$:

Update Flow
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Update Residual
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: capacity

$G_f$: Find Path

$G$: capacity

$G_f$: Find Path
Ford-Fulkerson Alg: Greedy on $G_f$

Update Flow

$G$:

$G_f$:
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: capacity

$G_f$: Update Residual
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Find Path
Ford-Fulkerson Alg: Greedy on $G_f$

**Update Flow**

$G:$

$G_f:$
Ford-Fulkerson Alg: Greedy on $G_f$

$G$:

$G_f$:

Update Residual
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Find Path
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Update Flow
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

```
G:  

s  10  

10  2  0  

10  2  2  8  

4  4  6  6  10

3  2  6  2  10

2  8  2  4  2

4  4  6  4  6

t  10
```

$G_f$: 

```
G_f:  

s  10  

8  2

10  2  2  7

4  4  6  4  6

3  2  2  2  6

5  7  2  4  4

4  6

10  4  2  6  6

t  10
```

Update Residual
Ford-Fulkerson Alg: Greedy on $G_f$
Ford-Fulkerson Alg: Greedy on $G_f$
Ford-Fulkerson Alg: Greedy on $G_f$
Ford-Fulkerson Alg: Greedy on $G_f$

$G$: 

$G_f$: 

Find Path
Ford-Fulkerson Alg: Greedy on $G_f$

Update FLow

$G:$

$G_f:$
Ford-Fulkerson Alg: Greedy on $G_f$

$G$:

$G_f$:

Find Path

capacity